Understanding fish behavior during typhoon events in real-life underwater environments

Spampinato, C., Palazzo, S., Boom, B., Ossenbruggen, J., Kavasidis, I., Salvo, R., Lin, F-P., Giordano, D., Hardman, L., & Fisher, R. B. (2012). Understanding fish behavior during typhoon events in real-life underwater environments. Multimedia Tools and Applications, 58(2), 1-38doi: 10.1007/s11042-012-1101-5

Understanding fish behavior during typhoon events in real-life underwater environments

Concetto Spampinato · Simone Palazzo · Bastian Boom · Jacco van Ossenbruggen · Isaak Kavasidis · Roberto Di Salvo · Fang-Pang Lin · Daniela Giordano · Lynda Hardman · Robert B. Fisher

Abstract The study of fish populations in their own natural environment is a task that has usually been tackled in invasive ways which inevitably influenced the behavior of the fish under observation. Recent projects involving the installation of permanent underwater cameras (e.g. the *Fish4Knowledge (F4K)* project, for the observation of Taiwan's coral reefs) allow to gather huge quantities of video data, without interfering with the observed environment, but at the same time require the development of automatic processing tools, since manual analysis would be impractical for such amounts of videos. Event detection is one of the most interesting

This research was funded by European Commission FP7 grant 257024, for the Fish4Knowledge project (www.fish4knowledge.eu).

C. Spampinato (✉) · S. Palazzo · I. Kavasidis · R. Di Salvo · D. Giordano
Department of Electrical, Electronics and Computer Engineering,
University of Catania, Catania, Italy
e-mail: cspampin@dieei.unict.it

S. Palazzo
e-mail: simone.palazzo@dieei.unict.it

I. Kavasidis
e-mail: isaak.kavasidis@dieei.unict.it

R. Di Salvo
e-mail: roberto.disalvo@dieei.unict.it

D. Giordano
e-mail: dgiordan@dieei.unict.it

B. Boom · R. B. Fisher
School of Informatics, University of Edinburgh, Edinburgh, UK

B. Boom
e-mail: bboom@inf.ed.ac.uk

R. B. Fisher
e-mail: rbf@inf.ed.ac.uk

aspects from the biologists' point of view, since it allows the analysis of fish activity during particular events, such as typhoons. In order to achieve this goal, in this paper we present an automatic video analysis approach for fish behavior understanding during typhoon events. The first step of the proposed system, therefore, involves the detection of "typhoon" events and it is based on video texture analysis and on classification by means of Support Vector Machines (SVM). As part of our behavior understanding efforts, trajectory extraction and clustering have been performed to study the differences in behavior when disruptive events happen. The integration of event detection with fish behavior understanding surpasses the idea of simply detecting events by low-level features analysis, as it supports the full semantic comprehension of interesting events.

Keywords Event detection · Fish detection · Covariance tracking · Behavior understanding

1 Introduction

The typical techniques adopted by marine biologists to study fish populations in their natural habitat involve casting nets in the ocean, human underwater observation and photography [48], combined net casting and acoustic (sonar) [6] and human hand-held video filming. However, these approaches suffer several limitations: for example, the net casting method, though accurate, has the disadvantage of killing the fish and damaging their environment; human filming and photography do not damage the habitat, but provide limited information.

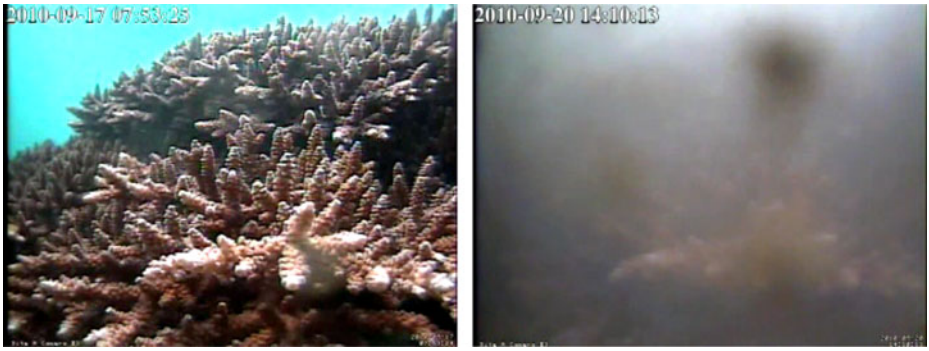
In order to overcome these limitations, in recent years the use of embedded underwater cameras has drawn a lot of interest, since it is an environmentally friendly approach, does not influence fish behavior and also provides large amounts of video material. On the other hand, it becomes impractical to manually analyze this huge quantity of video data, both because it requires a lot of time and concentration and also because it is error prone—it is unrealistic to assume people can fully investigate all the information in the videos. Therefore, automatic video analysis methods are heavily demanded.

J. van Ossenbruggen · L. Hardman
Centrum voor Wiskunde en Informatica (CWI),
University of Amsterdam, Amsterdam, Netherlands

J. van Ossenbruggen
e-mail: Jacco.van.Ossenbruggen@cwi.nl

L. Hardman
e-mail: lynda.hardman@cwi.nl

F.-P. Lin
National Center of High Performance Computing, Hsinchu City, Taiwan
e-mail: fplin@nchc.narl.org.tw



(a) Scene under standard conditions (b) Scene during a typhoon

Fig. 1 Underwater scenes recorded by the Coral Reef in Ken-Ting Area (Taiwan)

The Fish4Knowledge[1] project uses live video feeds from ten underwater cameras located in the coral reefs of Taiwan's shores and aims at developing an automatic system for integrated data capturing, video analysis, fish detection and classification, and querying, for the marine biologists to use, in order to study fish populations, behavior and interactions. The main difficulty of this kind of task is the nature of the videos to be processed. Traditionally, such tasks involve the analysis of video shot in controlled environments, such as tanks, where for example lighting conditions do not change with time, the background can be chosen to simplify fish detection, the type of fish is known, etc. The lack of these assumptions greatly complicates the task to be accomplished and requires the development of automatic analysis methods which are robust enough to handle all the possible varying conditions of the environment.

Another critical aspect of the project is the recognition of "events", both related to fish behavior (such as eating, preying, sleeping, mating, etc) and to environmental phenomena (such as typhoons), in order to have videos automatically labeled and categorized by the happening of such events. In fact, in marine biology it is of extreme importance to monitor habitat and fish fauna before and after catastrophic typhoons. For example, The Taiwan area is often hit by typhoons (e.g. five in 2010), which cause severe deleterious effects on fish and their habitat [33, 58]. Although there exist studies, e.g. [40] and [53], that have investigated the changes in underwater environment after such events happen, most of them have focused on long-term monitoring and they are not able to explore the effects right after the typhoon. This is mainly due to the impracticability for human operators to monitor the area because of, for instance, the floods coming down from neighboring mountains. Therefore, the importance of using underwater cameras (and intelligent video analysis approaches) rises again, especially for short/middle term monitoring but also for analyzing all those cases where a typhoon hits a specific area unexpectedly.

However, underwater camera-based monitoring shows some deficiencies (as above mentioned): the impossibility of a human operator to deeply analyze the stored videos and the poor clarity of the images during such events, that makes hardly possible not only the identification of fish behavior, but also of fish, as shown in Fig. 1.

[1] http://fish4knowledge.eu

In this paper we propose an approach that integrates typhoon event detection and fish behavior understanding based on the analysis of fish trajectory dynamics. Since the videos shot during with typhoons show an increase of the turbidity of the water, the event detection algorithm relies on classification of texture features of video frames by means of Support Vector Machines (SVM). Once a typhoon event is detected, fish trajectories are extracted by suitable detection and tracking modules in order to study to what extent typhoons influence fish behavior. Finally, to describe the differences in fish activities at the different times of a typhoon's passage, the clustering of fish trajectories in standard underwater conditions (i.e., no typhoon) and during typhoons is performed.

This paper focuses on typhoon event detection and trajectory-based behavior understanding (Section 4), which rely on automatic object detection and tracking methods. These are advances on previous underwater object detection, tracking and event analysis algorithms (Section 3).

The data for the project comes from the fixed camera underwater monitoring system of the Fish4Knowledge project (Section 2). We achieve an accuracy of about 80% on fish detection, 95% on tracking and 97% on event detection (Section 5). Possible improvements include (discussed in Section 6) the extension of the event detection algorithm to any event detection problem which can be formulated in terms of texture description of videos and the publication of all the achieved event data as Linked Open Data in order to provide sufficient detail to allow full replication of the experiments described, but also a sufficient level of abstraction to facilitate linking to existing event data.

2 The underwater monitoring system

The Fish4Knowledge project exploits the distributed system for real-time video streaming and storage provided by the National Center of High Performance Computing in Taiwan. This system includes a stream receiving unit, a stream processing unit and a presentation unit [10]. Figure 2 illustrates the distributed streaming system architecture and the stream pipeline.

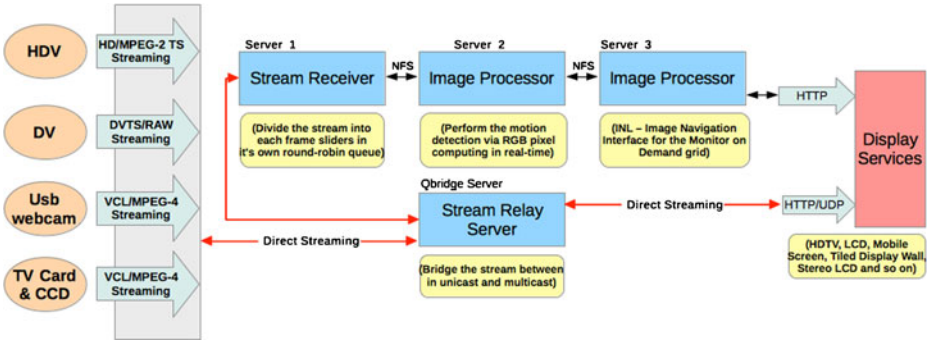


Fig. 2 Architecture blocks and stream pipeline

The stream receiving unit supports and identifies automatically multiple capture devices, choosing a proper encoder accordingly. It creates a device identification module, packed together with several commonly used software components into one package. This package contains not only the most commonly used devices, such as HDV, DV, DC, Webcams, TV Cards and Capture Cards, but also multiple stream compression encoders of the most commonly used standards, such as MPEG-1/2/4, WMV, FLV, MJPEG. The benefit of the architecture is the decoupling of the pipeline, which allows flexible configurations of capture devices and codec standards. The stream processing unit, after receiving a stream, offers two options: one that directly streams the received video to the presentation unit and another that slices it into a sequence of images, which can be extracted and stored for further processing. These images are sent to the respective round robin queues, to be treated by image processing techniques, such as event detection, object detection and tracking, image segmentation etc. When the image processing phase terminates, the images and the associated logging information are automatically stored in an image database. There, users can query, browse, analyze, and manage these images via image management services. The presentation unit supports multiple display devices to accommodate end-users' needs. The easiest way for users to view the streams is by using a Web browser. For this purpose, a Web-based user interface was designed to allow users to browse multiple real-time streams (Fig. 3). In order to reduce network traffic and to enhance the streaming efficiency, two distributed video stream compression concepts are implemented. The first distributed compression is a client-server based concept. It includes pre-compression on the client side and post-compression on the server side. The stream receiving unit grabs the video stream from the client and compresses it to reduce the necessary bandwidth needed. The post-compression transfers the

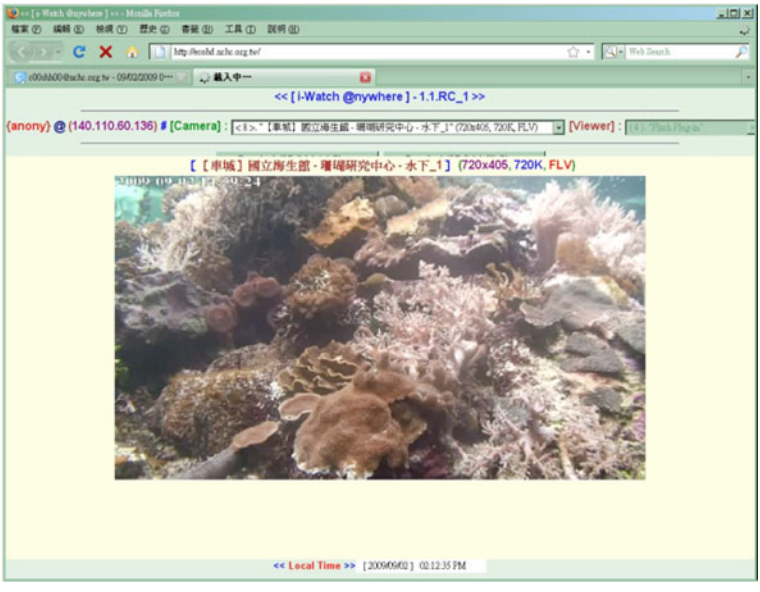


Fig. 3 A Web interface for stream viewing, users can select multiple real-time streams from *top panel*

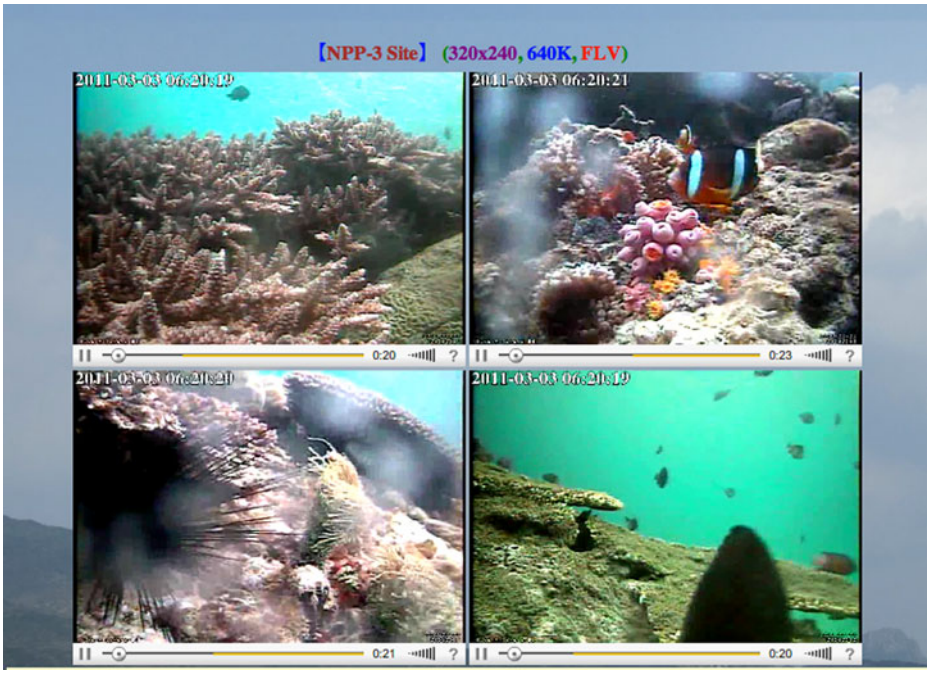


Fig. 4 An example of the coral reef underwater ecological observation

original video stream to multiple formats, such as MPEG-1/2/4, WMV, FLV and MJPEG, to reduce the video data volume [41, 65]. The second distributed compression is a server-side based concept. It dynamically assigns the video compression task to the most appropriate servers depending on their load. It compresses the video stream to a variety of bit-rates to suite different network bandwidth capabilities. A real-time high-resolution streaming system is implemented by combining these two distributed stream compression concepts. The main application of the described architecture is the long term underwater environment ecological observation for assisting marine ecologists to closely monitor the ecosystem of coral reefs in Ken-Ting National Park and in Orchid Island (Taiwan). Figure 4 shows an example of coral reef observation.

In detail, the underwater ecological observation system consists of ten cameras continuously recording during daylight. For convenience, the video stream is divided into ten min long video files at multiple resolutions (320×240 and 640×480) and at different frame rates ranging from 5 fps to 30 fps. Currently, the system contains videos of the last five years (2007–2011), i.e., each camera has recorded about 100,000 videos and therefore, the entire dataset for all the ten cameras is of about one million videos. The videos are available at the Web-link http://gad240.nchc.org.tw/.

3 Related works

In this paper we propose an automatic system for typhoon event detection that, differently from the existing approaches, integrates low-level events (fish trajec-

tories) with high-level events (typhoon), in order to understand how fish behave during specific events and thus supporting the full semantic comprehension of a specific event [51]. Therefore, one of the main parts of the proposed system is fish trajectory analysis, which involves fish detection and tracking. For this reason, at the beginning of this section, a brief review of the detection and tracking methods that work under conditions similar to the underwater's ones is presented; then, the existing approaches in underwater domain are reviewed. Finally, the analysis of the existing approaches for event detection in underwater scenes is given.

Before reviewing the literature, let us introduce which are the effects that usually occur in underwater scenes and that make the task of video analysis very difficult and challenging:

- ***sudden*** and ***gradual*** **light changes**: typically, the videos are available starting from sunrise up to sunset, so it is necessary to consider the light transition due to these particular moments of the day in which brightness and contrast of the images are strongly compromised by the absence of sunlight. Moreover, the periodical gleaming in the underwater scene has to be considered when designing the detection and tracking algorithms;
- **bad weather conditions**: the weather conditions could be subject to unexpected changes such as sudden cloudiness, storms and typhoons. Under these conditions the scene becomes very difficult to analyze due to a worsening of image contrast which makes it hard to detect and track clearly any targets;
- **murky water**: in order to investigate the movements of fish in their natural habitat it is important to consider that the clarity of the water during the day could change due to the drift and the presence of plankton. Under these conditions, targets that are not fish might be detected as false positives;
- **algae** on **camera lens**: the direct contact of seawater with the lens causes a rapid formation of algae and filth that compromises the quality of the observed scene;
- **periodic** and **multimodal background**: handling background movements and variations is one of the most difficult tasks and algorithms must be robust enough to cope with any arbitrary changes in the scene. Also periodic movements (e.g. plants affected by flood-tide and drift) have to be taken into account to avoid the detection of moving non-fish objects;

However, one of the most complex issues to deal with, when processing underwater videos, concerns the targets to be detected and tracked. Indeed, differently from humans, fish show erratic and fast movements (in three dimensions) that, therefore, lead to frequently changes in size and appearance.

All these aspects can be classified as extreme conditions, and in the following the object detection and tracking methods that deal with similar conditions are reviewed.

3.1 Object detection ad tracking under extreme conditions

A myriad of detection algorithms have been proposed for handling different backgrounds and scene phenomena. Basically, the existing approaches for motion modeling [9, 19] can be classified into recursive techniques [18, 62] which adaptively update single or multiple background models based on each input frame at the current time, and non-recursive techniques [22, 64] that use a buffer of N previous frames to estimate the background image according to the temporal variation of

each pixel within the buffer. However, none of these approaches have demonstrated to be generally superior to the other ones, and the performance depends on the specific application domain. Porikli in [44] have compared different algorithms (both for detection and tracking) under extreme conditions that somehow recall the ones present in underwater scenes such as erratic motion, sudden and global light change, presence of periodic and multimodal background, arbitrary changes in the observed scene, low contrast and noise. In detail, the authors have shown that the algorithms that best perform under these conditions use mixture of probability density function (pdf) models [23, 24, 62], Wave-Back Model [45] and Intrinsic Model [43]. In detail, mixture of pdf models have been adopted to handle multimodal backgrounds. In these models, background pixels are modeled as a mixture of either Gaussian or Poisson pdfs, which are iteratively updated at every frame. These approaches are flexible enough to handle sudden and global illumination changes and other arbitrary variations in the scene and, moreover, they can converge to any arbitrary distribution providing enough number of observations, but generally the computational cost grows exponentially as the number of models in the mixture increases. A drawback of the previous methods is that they ignore the temporal correlation of color values. This impedes the differentiation of periodic background motion from foreground motion. Since real-world physics (especially, in underwater domains) induces near-periodic phenomenon in the environment, a frequency decomposition-based representation of the background (Wave-Back [45]) has been also adopted. This algorithm detects moving objects based on the form of the temporal color variation by comparing the frequency transform responses. Basically, the algorithms use frequency decomposition of pixels' history to catch periodic background movements. In particular, for any input frame the DCT (Discrete Cosine Transform) coefficients are calculated and compared to the respective background coefficients, thus resulting in a distance map. By thresholding this distance map, the moving objects can be isolated. The Wave-Back algorithm usually performs well in repetitive scenes and with low-contrast colors but it performs inadequately in scenes with erratic movement and when sudden lighting transitions take place.

To overcome these limits, background scene has been represented using intrinsic images [43] as a multiplication of static and dynamic parts. The idea behind this algorithm is that in a video, every image can be decomposed in two components: the reflectance image and the illumination image. The former is invariant to lighting changes and is almost identical under any light conditions. The background is modeled by calculating the temporal median of these reflectance parts of the input images. This algorithm has been proved to perform better than the previous ones under extreme conditions, specially, in scenes with lighting changes, fast-and-erratic object movements and low-contrast.

In a behaviour understanding system, tracking is of core importance since it extracts the trajectories of the objects involved in the scene and, at the same, it allows to repair detection failures. Visual tracking consists of following an object in a video across consecutive frames; in other words a tracking algorithm has to be able to recognize that two regions in two different frames represent the same object. However, this task presents a few major difficulties, which are even more emphasized in unconstrained environments such as the underwater one. First of all, in a tracking algorithm, the search region of an object has to be limited to a

neighborhood of its previous detection, otherwise a new object appearing in the scene might be associated to the old one, even if it is located in a different part of the video. The choice of the search area depends on the motion characteristics of the objects which typically appear in the scene. For example, in an urban environment we can safely assume that pedestrians and cars (e.g. the same holds for football players) move approximately always in the same direction, without swift changes, and this restricts the search area to something shaped like a cone oriented towards the main direction of the target. However, with fish this is not necessarily true, because their typical erratic motion in three dimensions makes their direction less predictable. Another tracking issue consists of the change in appearance of an object across the video, because of variations of lighting, orientation, shape. This is especially true for fish, because of their non-rigidity and, again, of erratic motion. Finally, another complication is caused by occlusions, that is the case of partial or total overlapping of two or more objects.

Many different approaches have been studied in literature on how to solve the visual tracking problem. Among these, the most famous and widely-used algorithms [32] are Kalman filter-based tracking [16], multiple hypothesis tracking (MHT) [47], optical flow-based tracking [56], particle filter tracking [27], point feature tracking, and mean shift tracking [12]. A detailed analysis and comparison of the existing approaches is beyond the aim of the paper, and an extensive literature of object tracking approaches may be found in [70] and [49]. However, a brief description of the existing approaches showing pros and cons under extreme conditions is, here, given.

One of the simplest ways to see the tracking problem is as an estimation of the probability density function of a state representing an object's position and appearance, given the set of all measurements up to that moment. When the measurement noise is assumed to be Gaussian, Kalman filters provide an optimal solution. The most general class of such filters is represented by particle filters, where the current state distribution is modeled as a set of weighted samples which are updated as soon as new measurements become available. Particle filter tracking is used in several applications [5, 71], however it may become impractical because of the size of the state vector and of the large number of particles in complex scenes (such as underwater domain). Another tracking approach consists in characterizing objects by local point features [37, 56] and trying to match objects through frames by evaluating the correspondences between feature point sets, chosen in such a way as to make the description of the objects invariant to affine or projective transformations. However, this technique presents several limits, especially with smooth object surfaces (for which it is difficult to extract distinguishing feature points) or when objects undergo pose changes, intersections and severe deformations as often happens with fish.

A computationally efficient and very popular approach is mean-shift tracking [12], which models the object's probability density in terms of color histogram, and moves the object region towards the largest gradient direction, in order to maximize the similarity between the reference and candidate object regions, measured with the Bhattacharyya coefficient or the Kullback-Leibler divergence criteria. However, this technique fails in the case of occlusions and quick appearance changes, when the color distribution of the background is too similar to that of the target object or when the object moves outside of the kernel search area. Tracking algorithms

based on covariance representation [66] model objects as the covariance matrices of a set of feature built out of each pixel belonging to the object's region. This representation embodies both the spatial and statistical properties, unlike from histogram representations (which disregard the structural arrangement of pixels) and appearance models (which ignore statistical properties) and, therefore, it is the most suitable to track objects under the above described conditions.

3.2 Object detection and tracking in underwater scenes

The existing literature for automatic video and image analysis of real-life underwater environments has been mainly focused on fish recognition [3, 34, 39, 61], whereas only few approaches for fish detection and tracking have been proposed.

Most of the existing methods have largely been driven by commercial fish aquaculture, with the goal of non-intrusive estimation of fish numbers and sizes in controlled environments or in labs (i.e., with fixed lighting, cameras, background, fixed objects in the water, known types of fish, known number of fish, etc.) in order to provide also useful feedback for the study of behavioral and locomotion under different environmental variations.

The first step before detecting and tracking fish is to perform image registration and to improve the quality of the grabbed frames. In this direction very few approaches have been proposed in underwater environment: for image registration, recently, Costa et al. [13] developed an approach based on artificial neural networks for correcting the distortion due to camera lens (as far as we know no approach exists for correcting jitter). Iqbal et al. [30] developed an enhancement system based on contrast stretching for solving lighting problems or clarity of water problems. An extensive literature of underwater image enhancement and restoration methods can be found in [52].

A variety of methods for fish detection and tracking have been proposed. Morais et al. [38] proposed a system, based on Bayesian filtering techniques, to detect and count fish in a fish tank with a fixed number of fish, reporting a success rate for fish counting of 81%. Evans [21] detected and counted isolated Southern Bluefin tuna in cages. Spampinato et al. [60], instead, proposed a system that detects and counts live fish free swimming by the coral reefs through the use of a video change detection algorithm with a performance of about 85%. Zhou and Clark [72] tracked individual Large Mouth Bass through multiple frames while simultaneously estimating their 3D position and orientation. Walther et al. [69] developed an automatic machine vision system for animal detection and tracking by using high-resolution video equipment on board of the ROV (ocean-going remotely operated vehicles—ROV). Hariharakrishnan and Schonfeld [28] proposed a tracking system based on the prediction of object contour by analyzing motion vector information.

3.3 Event detection in underwater scenes

In the last few years, high-level event detection and description in videos has drawn increasing interest from the scientific community [2]. Video surveillance systems, advanced human-computer interfaces and semantic video indexing are just the main examples of the possible application fields of this research branch. However, the video event recognition task has often been tackled in a domain-specific way (except

few exceptions, such as [14, 42])—where the typical domains are for example sports, movies, video-surveillance and user-generated content—which caused the lack of a general detection framework [2]. The most popular application fields for event detection in videos are sports [36], video-surveillance [1], road traffic control [55].

Event detection and, more in general video data analysis approaches have rarely dealt with effective real-life environments as those concerning the eco-system monitoring, which involves different and challenging domains with respect to common domains with humans [31, 57], since animals, fish and insects have more degrees of freedom than humans moving in a scene.

To best of our knowledge only a little number of approaches have dealt with event detection on unconstrained underwater scenes. In particular, Edgington et al. [17] proposed a system operating on the deep ocean where interesting low-level events (i.e. rare animals) are identified by using a model for saliency-based attention in humans and then such events are tracked. Similarly, Cline et al. [11] developed a neuromorphic vision approach for low-level event detection in the deep ocean. One approach that, instead, focuses on high-level events mainly related to group fish movement has been proposed in [59], where an automatic system for crowd flow analysis in underwater scenes was used to investigate fish schooling characteristics. The approach exploits Lagrangian particle dynamics from fluid mechanics in order to examine the trajectories of small particles in the fish flow and achieved satisfying performance in detecting events, such as fish schooling as a group flow.

4 The proposed system

The proposed typhoon event detection is based on texture analysis (at frame level) and on Support Vector Machines (SVM). This system is then integrated with a trajectory-based behavior analysis to fully understand the semantics of the event we are dealing with. Figure 5 depicts the flowchart of the whole system: in detail, a video is input into the typhoon detection and to the fish detection/tracking blocks. The former extracts a subset of the frames from the video and for each of them computes a set of features representing the texture characteristics of the image. In order to reduce the size of these vectors, Principal Component Analysis is performed and the resulting features are sent to an SVM classifier, which has been trained from ground-truth data to classify between *typhoon* and *non-typhoon* videos. Independently, fish detection and tracking algorithms are run and the results are sent to a trajectory analysis block, which builds clusters of similar trajectories; by joining these results with the output of the typhoon detection subsystem, it is possible to draw some conclusions on the behavior of fish in the periods before, during and after typhoons.

In the following sections, the single blocks, shown in the flowchart, are described in detail.

4.1 Typhoon event detection

The aim of this module is to identify, among all the recorded videos, the ones that show a typhoon event. By analyzing all the available videos, it is possible to notice that the best discriminant for typhoon event detection is the turbidity of the water, that is related to the image texture. Figure 1 shows how radically the global video

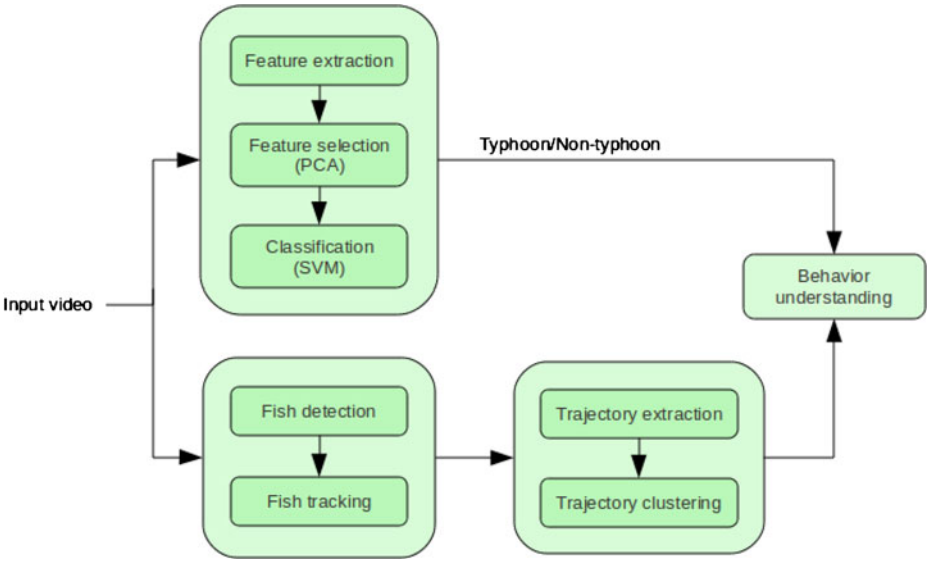


Fig. 5 Flow chart of the proposed typhoon detection and behavior understanding system

appearance changes after the arrival of a typhoon in the area. Accordingly, the detection of typhoon events is based on computing blurring and texture features on different parts of the images and at different scales. The features extraction method takes inspiration from the object detection algorithm described in [68], where the model of the background is built by iteratively dividing the image into a grid, computing histogram-based features on each cell, grouping cells into coarser blocks and repeating the procedure, until the size of the cells becomes too large (according to user criteria). The subdivision of the input image into a grid provides a convenient representation for both global and local texture statistics. The main disadvantage of this approach is the computation time, since the number of sub-images to process can be very large, because of the iterative grid division algorithm. Nevertheless, it is not necessary to process every frame in a video, since the global appearance of the scene is unlikely to change suddenly. For this reason, we only process a subset of frames (per video) in order to achieve the best compromise between computation costs and variability of the recorded scene. The proposed algorithm (the flowchart of which is depicted in Fig. 6) for feature extraction is described in Algorithm 1. The initial cell size is set to 40×30 pixels in order to keep the 4:3 ratio of the input videos.

The output vector describing global and local texture contains a number of elements that depends on the video resolution; for 320×240 videos, it stores 3,910 values for each frame. In order to reduce the dimensionality of the problem, principal component analysis (PCA) is then applied for features selection. The data obtained is then fed to a Support Vector Machine with a quadratic kernel, because of their good generalization property [29], as a binary classifier for identifying videos that depict typhoons from ones that do not. After typhoon event detection is accomplished, fish detection and tracking is performed to extract the trajectories used for investigating fish behavior during typhoons.

Algorithm 1 Pseudo-code implementation of the feature extraction algorithm

```
Initialize feature vector
featureVector ← ∅
Initialize cell size
cellSize.height ← minimumCellHeight
cellSize.width ← minimumCellWidth
Divide the image into a grid with the initial cell size
cells ← image.divideInCells(cellSize)
Start feature computation loop
while cells.size() ≥ minimumGridSize do
  for cell in cells do
    Get RGB and grayscale histograms
    redChannelHistogram ← cell.getRedChannelHistogram()
    greenChannelHistogram ← cell.getRedChannelHistogram()
    blueChannelHistogram ← cell.getRedChannelHistogram()
    grayHistogram ← cell.toGrayscale().getRedChannelHistogram()
    Add histogram-related features
    featureVector.add(redChannelHistogram.mean())
    featureVector.add(redChannelHistogram.variance())
    featureVector.add(redChannelHistogram.entropy())
    featureVector.add(redChannelHistogram.uniformity())
    ...
    featureVector.add(grayHistogram.mean())
    featureVector.add(grayHistogram.variance())
    featureVector.add(grayHistogram.entropy())
    featureVector.add(grayHistogram.uniformity())
    Apply Gabor filter at several scales and orientations
    for scale in 2, 4, 6, 8 do
      for orientation in 0, π/4, π/2, 3π/4 do
        gaborOut ← cell.applyGaborFilter(scale, orientation)
        Add Gabor-related features
        featureVector.add(gaborOut.mean())
        featureVector.add(gaborOut.variance())
      end for
    end for
    Compute gray-level co-occurrence matrix
    gclm ← cell.computeCoOccurrenceMatrix()
    Add co-occurrence-related features
    featureVector.add(gclm.maximumProbability())
    featureVector.add(gclm.elementDifferenceMomentOfOrder(2))
    featureVector.add(gclm.elementDifferenceMomentOfOrder(3))
    featureVector.add(gclm.inverseElementDifferenceMomentOfOrder(2))
    featureVector.add(gclm.inverseElementDifferenceMomentOfOrder(3))
    featureVector.add(gclm.entropy())
    featureVector.add(gclm.uniformity())
  end for
  Increase cell size
  cellSize.height ← cellSize.height × 2
  cellSize.width ← cellSize.width × 2
end while
Reduce the dimensionality by applying PCA
featureVector ← PCA(featureVector)
```

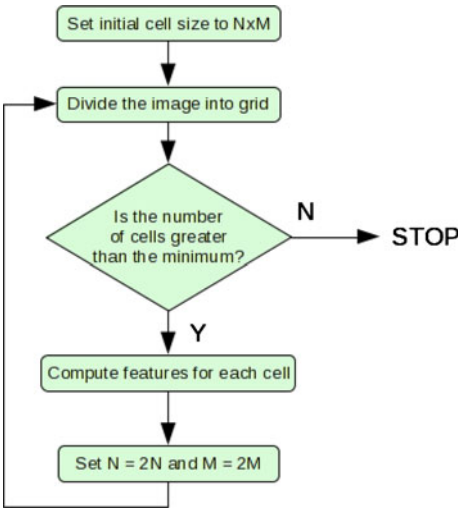


Fig. 6 Flow chart describing the algorithm for the extraction of features from a video

4.2 Fish trajectory extraction

4.2.1 Fish detection

Following the evaluation on detection algorithms under extreme conditions performed by Porikli [44], we have implemented four approaches for fish detection, namely, Adaptive Gaussian Mixture Model ($AGMM$), Adaptive Poisson Mixture Model ($APMM$), Intrinsic Model (IM) and Wave-Back (WB), each one dealing with some aspects of the underwater scenes as described in Section 3. These algorithms perform fairly well (as shown in the experimental result section) in underwater environment, however they often detect non-fish objects (i.e. false positives) that have to be filtered out. This requirement led us to provide the fish detector with an additional level of post-processing that assigns to each detected blob a quality score. This score is a numerical value between 0 and 1 and is computed as the average of the following features:

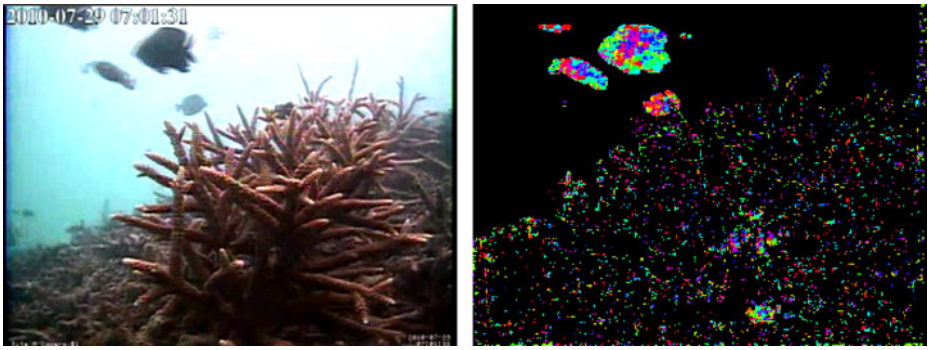
- *Difference of color at object boundary* (Δ_{CO}): this index is based on the assumption that color boundaries often coincide with fish/background boundaries; this value is highest when the areas "just inside" and "just outside" of the contour of the detected fish have markedly different color values. It is computed according to the following procedure.
  - Select N equidistant points on the object's contour. The current value for N (the number of sampled contour points) has been set to 25. This choice is due to the fact that the minimum fish contour size, in pixels, has been estimated to be about 150–160 points for 320×240 images. Sampling one point in every six allows to accurately follow the contour of the object, without missing sharp changes in the shape.

– For each point P_i,
  - Select two points $P_{i,in}$ and $P_{i,out}$ located just inside and just outside of the contour on the line passing by P_i and such that it is orthogonal to the tangent of the contour in that point.
  - Compute $C_{i,in}$ and $C_{i,out}$ as the average color intensities in the 5×5 neighborhoods of $P_{i,in}$ and $P_{i,out}$.
– Compute the result as:

$$\Delta_{CO} = \frac{1}{N} \sum_{i=1}^{N} \frac{\| C_{i,in} - C_{i,out} \|}{\sqrt{3 \cdot \alpha^2}} \tag{1}$$

In this formula, the numerator of the fraction inside the sum is the norm of the color intensity between the pair of pixels, and the denominator represents the minimum Euclidean distance between two pixels to be considered as belonging to markedly separate color regions. Ideally, for each pair of points, the contribution to the sum should be low if the contour belongs to a homogenous region, and high if the color difference is greater than the reference distance (the denumerator).

– *Difference of motion vectors at object boundary* Δ_{MV}: similarly to the previous case, the motion vector in the regions close to the object contour are compared; this value (computed by formula (2)) takes into account the fact that, hypothetically, the motion vector outside of the object contour should be zero (static background) or significantly different than inside (as shown in Fig. 7). In the following formula, $M_{i,in}$ and $M_{i,out}$ represent the average motion vectors



(a) Original frame (b) Motion vector

Fig. 7 Motion vector boundaries typically coincide with object boundaries

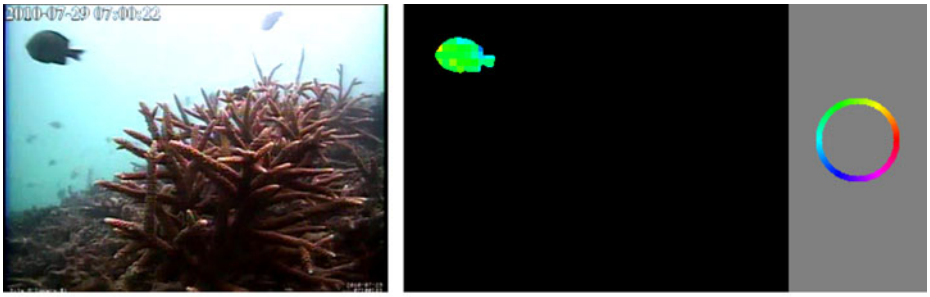
computed just inside and just outside of contour point P_i (obtained as in the previous formula).

$$\Delta_{MV} = \frac{1}{N}\sum_{i=1}^{N}\frac{||M_{i,in} - M_{i,out}||}{||M_{i,in}|| + ||M_{i,out}||} \tag{2}$$

As in the previous formula, this score returns a high value if the motion vectors are different, and a low value otherwise. The denumerator allows to normalize between 0 and 1, since the norm of the difference is necessarily smaller than the sum of the respective norms.

- *Internal color homogeneity*: due to the low resolution of videos, most fish (especially the ones far away from the camera) appear as monochromatic, so this index gives an indication on how homogenous the color distribution of the detected fish is. The body of the object is divided into a grid and for each cell the average color value is computed; the more similar these average results are, the more likely it is that the detected object is actually a fish. This value is computed as follows:
  - For each cell j in the grid, compute the mean color C_j.
  - Compute C_M as the mean of all $\{C_j\}$.
  - For each C_j, compute $d_j = ||C_j - C_M||$.
  - Compute d_M as the mean of all $\{d_j\}$.
  - Return $-\beta d_M + \gamma$ (this score decreases linearly as the average difference between each cell's color and the average cell color increases).
- *Internal motion homogeneity*. This index is based on the assumption that the internal motion vectors of a correctly-detected fish are more uniform than the ones of an erroneous fish detection, as shown in Fig. 8. Similarly to the previous case, the object is divided into a grid and the average motion vectors for each cell are compared. However, the computation of the corresponding score is slightly different:
  - Given an object, its current bounding box and the bounding box of its last appearance, compute the motion vector of the region R obtained as the union of the two bounding boxes.
  - For each motion vector point in R, mark it as "not valid" if its displacement projects the point out of R. This might happen because the motion vector algorithm is not accurate for points belonging to the common region between two detections.
  - For each cell j in the grid which has at least one valid motion vector point, compute the mean motion vector V_j, whose components are the average Δx and Δy.
  - Compute the variances v_x and v_y of the first and second components of all $\{V_j\}$.
  - Return $1 - \frac{v_x + v_y}{\delta}$.

The parameters α, β, γ and δ are used to normalize the quality score in the range [0–1], and their computation is specified in Section 5. For the investigation of fish



(a) Original frame (good detection) (b) Filtered motion vector (good detection)

(c) Original frame (bad detection) (d) Filtered motion vector (bad detection)

Fig. 8 Correct object detections are made up of points generally orientated towards the same direction. The *circle* to the right of subfigures **b** and **d** maps color to direction of motion

behavior during typhoons, we consider fish to be all those objects with a quality score larger than 0.8. After the fish detection, the tracking algorithm is carried out, which serves both as a proof of the correctness of the detected fish—i.e., a moving fish usually remains in the monitored scene longer than a spurious object and its movements (as erratic as possible) are more regular than plants' ones—and to extract consistently fish paths necessary for the trajectory-based behavior understanding module.

4.2.2 Fish tracking

The tracking algorithm, adopted to handle all the phenomena typical of underwater domain, is based on [46] and uses covariance matrices (since the covariance-based tracker has been demonstrated to be superior under extreme conditions [44]) computed on a set of pixel-based features to model the fish appearance. In the following description, we use "tracked object" to indicate an entity that represents a unique fish and contains information about the fish appearance history and its current covariance model; and "detected object" to indicate a moving object, which has not been associated to any tracked object yet. For each detected object, the corresponding covariance matrix is computed by first building a feature vector for each pixel, made up of the pixel coordinates, the RGB and hue values and the mean and standard deviation of the grey level histogram of a 5×5 window which contains the target pixel.

The covariance matrix, which models the object, is then computed from this feature vector and associated to the detected object. Afterwards, the covariance matrix is used to compare the object with the currently tracked objects, in order to decide which one it resembles the most. The main issue in comparing covariance matrices is that they do not lie on the Euclidean space—for example, the covariance space is not closed under multiplication with negative scales. For this reason, as suggested in [46], we used Förstner's distance [25], which is based on generalized eigenvalues, to compute the similarity between two covariance matrices:

$$\rho\left(C_i, C_j\right) = \sqrt{\sum_{k=1}^{d} ln^2 \lambda_k \left(C_i, C_j\right)} \tag{3}$$

where d is the order of the matrices and $\left\{\lambda_k \left(C_i, C_j\right)\right\}$ are the generalized eigenvalues of covariance matrices C_i and C_j, computed from

$$\lambda_k C_i x_k - C_j x_k = 0 \quad k = 1 \cdots d \tag{4}$$

The model of each tracked object is then computed as a mean of the covariance matrices corresponding to the most recent detections of that object. In order to deal with occlusions, the algorithm handles the temporary loss of tracked objects, by keeping for each of them a counter for how many frames it has been missing; when this counter reaches a user-defined value (*time-to-live*, TTL), the object is considered lost and discarded. Of course, it is important to find a good trade-off for this value: if it is too low, an object which temporarily disappears (for example, because it is hidden behind a plant) might be treated as a different object when it reappears; if it is too high, different objects might be recognized as the same one, which has long exited the scene.

The steps performed by the proposed tracking algorithm are described in detail in Algorithm 2.

The fish tracking algorithm is one of the most important parts of the proposed system since it identifies the fish trajectories on which the behavior understanding module relies. Thereafter, in order to consistently investigate fish behavior, it is necessary to estimate the quality of each detected trajectory and to select the ones that respect a specific criteria of goodness, thus avoiding to invalidate the final behavior analysis. To compute such a quality score (referred in the following as q_S), we have adopted a series of measurements taken from [8] and [20] and combined them with new measurements in order to obtain values indicating the goodness and the feasibility of a trajectory. In detail, for each tracking decision (i.e. an association between an object in frame t and one in frame $t+1$) the following features are computed and combined into a single score as an average. As for the detection quality scores, the formulas below are parameterized by normalization constants (which are described in Section 5).

– *Difference of shape ratio between frames*: this score detects rapid changes in the object's shape, which might indicate tracking failure. This value is high if the shape ratio ($R = \frac{W}{H}$, with W and H, respectively, the width and the height of

Algorithm 2 Pseudo-code implementation of the tracking algorithm

```
detected_objects ← runDetectionAlgorithm()
tracked_objects ← getCurrentlyTrackedObjects()
feasible_associations ← ∅
Compute covariance matrix for each detected object
for D_j in detected_objects do
    D_j.computeCovarianceMatrix()
end for
Compute set of possible associations between tracked objects and detected objects
for T_i in tracked_objects do
    for D_j in detected_objects do
        if T_i.getCurrentPosition() ∩ D_j.getBlob() ≠ ∅ then
            d_ij ← computeCovarianceDistance(T_i.currentModel(),
            D_j.covarianceMatrix())
            feasible_associations.add((T_i, D_j, d_ij))
        end if
    end for
end for
sortByDistance(feasible_associations)
Assign each detected object to the covariance-closest tracked object
for (T_i, D_j, d_ij) in feasible_associations do
    if not D_j.isAssigned() then
        D_j.assignTo(T_i)
        T_i.updateModel()
    end if
end for
If a tracked object has been missing for too many frames, remove it
for T_i in tracked_objects do
    if T_i.foundInCurrentFrame() then
        T_i.resetTTL()
    else
        T_i.decreaseTTL()
        if T_i.getTTL() = 0 then
            tracked_objects.remove(T_i)
        end if
    end if
end for
Add new tracked objects
for D_j in detected_objects do
    if not D_j.isAssigned() then
        tracked_objects.createNew(D_j)
    end if
end for
```

the bounding box containing the object) between consecutive frames $t-1$ and t keeps as constant as possible:

$$R_{\max} = \max\{R_t, R_{t-1}\}$$

$$R_{\min} = \min\{R_t, R_{t-1}\}$$

$$shape\_ratio\_score = \frac{R_{\min}}{R_{\max}}$$

– *Histogram difference*: this feature evaluates the difference between two appearances of the same object by comparing the respective histograms (analyzing independently the three RGB channels and the grayscale versions of the two objects). Given histograms H_t and H_{t-1}, the corresponding score is computed by subtracting each bin's difference (scaled by a constant) from the maximum score (1):

$$1 - \frac{1}{\epsilon} \sum_{i=0}^{255} |H_t(i) - H_{t-1}(i)|$$

– *Direction smoothness*: assuming a trajectory is as good as it is regular and without sudden direction changes, this value keeps track of the direction of the object in the last frames and checks for unlikely changes in the trajectory. It is computed as:

$$direction\_smoothness = 1 - \frac{|\theta_1 - \theta_2|}{180}$$

where θ_1 and θ_2 are the angles (with respect to the x axis) of the last two displacements of the object. For simplicity, we use $\theta_1 - \theta_2$ in the formula, although the actual implementation handles the case of angles around the 0°/360° boundary. According to this formula, the corresponding score is equal to 1 if the object's direction keeps constant in two consecutive frames ($\theta_1 = \theta_2$), and is 0 if the object's direction is inverted.

– *Speed smoothness*: similarly to the previous feature, this value checks whether the current speed of the object (i.e. the displacement between the previous position and current one) is similar to the average speed in the object's history. Let P_t and P_{t-1} be the last two positions of the object, we compute $s_t = ||P_t - P_{t-1}||$, so that s_t represents the last displacement (speed) of the object, and compare it with the average speed $\bar{s}$ in order to compute *speed_smoothness* as:

$$s_{\max} = \max\{s_t, \bar{s}\}$$

$$s_{\min} = \min\{s_t, \bar{s}\}$$

$$speed\_smoothness = \eta \frac{s_{\min}}{s_{\max}}$$

– *Texture difference*: texture features (mean and variance of several Gabor filters) are computed from two consecutive appearances and compared. Given two feature vectors v_1 and v_2, this value is computed by subtracting the (scaled) Euclidean distance from 1:

$$1 - \sqrt{\sum_{i=1}^{n} (v_1(i) - v_2(i))^2}/\lambda \tag{5}$$

– *Temporal persistence*: this value is the number of frames in which a given object appears.

The overall quality score q_S is computed as follows:

– Compute the average μ of the above-described values, except the temporal persistence TP.

(a) Unusual fish trajectory (b) Feasible fish trajectory

Fig. 9 Some results about fish tracking: **a** an erroneous path (average q_S score 0.63) due to a fail of the tracking algorithm, **b** a correct path (average q_S score is of 0.91)

- If $TP > 5$, return μ;
- Else, return $\mu \cdot \left(0.9 + \frac{TP}{50}\right)$.

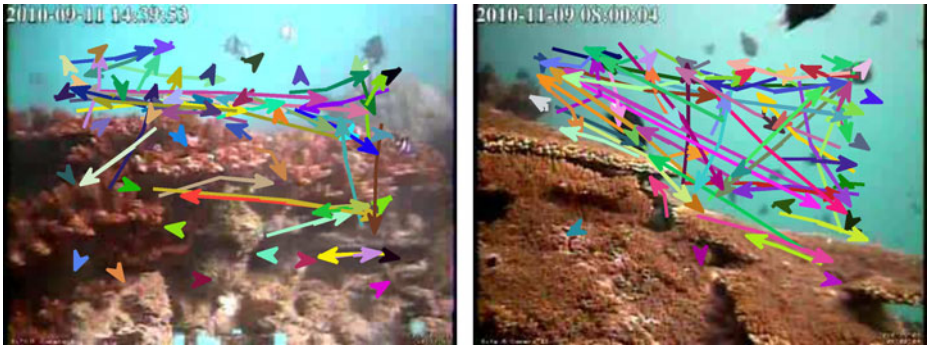
In the formula above, the *temporal persistence* score is used to limit the actual maximum overall score achievable: if the object has appeared for less than 5 frames, then the maximum score is limited, proportionally, between 0.95 and 1.

Figure 9 shows two sample trajectories with related average q_S scores; it is possible to notice that the trajectory of the left image is unrealistic and its average score, computed as average of the scores of each tracking decision for all the appearances of a fish (four times, in this case), was of 0.63, whereas the image on the right side shows a correct trajectory whose average score is of 0.91.

4.3 Fish trajectory analysis

In marine biology, it is interesting to observe how the occurrence of anomalous events can influence fish behavior. In this section we describe a trajectory-based behavior understanding approach for investigating how fish act when typhoons happen. In detail, the behavior understanding module uses fish trajectories to describe usual fish habits and, based on our knowledge of the periods when typhoons occur, allows us to study the behavioral deviations that might apperar during such events.

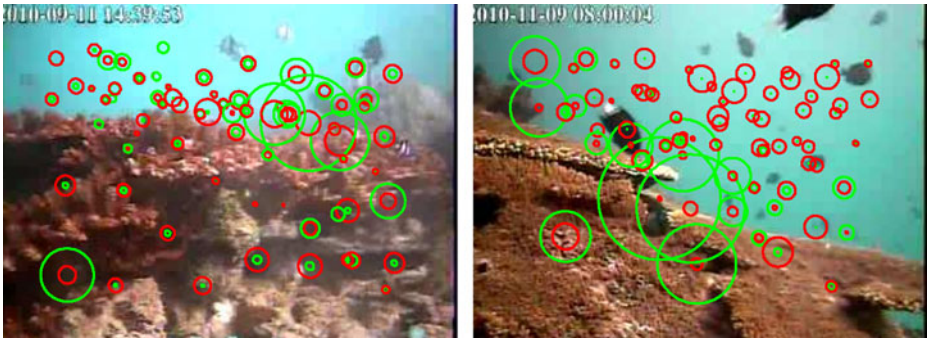
The combined information of fish detection and tracking gives us the apparent trajectories (due to the 2D nature of the input data) that fish follow in the videos. There exist different ways to represent trajectories, although the literature in this area focuses mainly on video surveillance applications [31, 57]. However, in underwater recordings fish have more degrees of freedom than humans walking in a scene; furthermore, fish sometimes tend to stay in one place instead of following clear trajectories. In order to model 3D fish paths, a representation based on the x, y location and blob size of the fish, which gives an indication of the depth z, is used. Therefore, for each appearance of a fish we keep the tuple $T = (x, y, z)$ and a path is represented by its start, middle and end locations $(T_{\text{first}}, T_{\text{middle}}, T_{\text{end}})$. Given this representation, we divide the set of all trajectories into fish trajectories $P = \{p_i\}_{i=1}^{N}$ during standard underwater conditions and fish trajectories during typhoons $Q =$



(a) NPP3 camera 1 (b) NPP3 camera 2

Fig. 10 The fish trajectories computed from the recordings of two underwater cameras in Kenting, Taiwan (http://ecosite.nchc.org.tw/kenting/). *Camera 1* shows how fish tend to hide in the coral's holes (as indicated by the *arrows* close to the holes), whereas above the coral, fish seem to swim more in paths. In *Camera 2*, a coral without much caves is shown, where fish tend to swim along the coral or just horizontally into the open sea

$\{Q_j\}_{j=1}^{M}$. Afterwards, an unsupervised k-means clustering [63] is applied on the P and Q sets, thus achieving a set C of clusters. Each obtained cluster is represented by its centroid, which is still a trajectory. Figure 10 depicts the achieved centroids as the average paths on a sample set of videos (the same as the one used for the experimental results). Unlike what generally happens in the video-surveillance fields, we can observe that fish often do not have well defined paths, due to their erratic movements and frequent changes in size and appearance. Once the k-means algorithm is applied and all the clusters are computed, the percentage of trajectories p_T that belong to each cluster can be derived. Figure 11 shows the p_T values for all the clusters shown in Fig. 10 both for trajectories P (red circles) and for trajectories



(a) NPP3 camera 1 (b) NPP3 camera 2

Fig. 11 These images show *red* and *green circles* around the clusters' centroids; the radius of the *red circles* indicates the percentage of trajectories that fall into a cluster when no typhoons occur, while the radius of the *green circles* is the percentage of trajectories followed during a typhoon. It is clearly visible that in the former case the trajectories are almost evenly distributed, while during a typhoon fish prefer certain trajectories. In *camera 2*, it seems also that fish do not go into the open sea and stay close to the coral

Q (green circles). The bigger the radius of a circle is, the more trajectories have been assigned to the cluster found in that specific point. Therefore, the size of the red circles indicates how often a certain trajectory is followed by fish when no typhoon affects the recorded scene, whereas the size of the green circles indicates the percentage of followed trajectories by fish during typhoons. It is possible to notice how, in the former case, the distribution of the clusters is quite regular over the scene, while in the latter case, the trajectories are not evenly distributed anymore.

Moreover, during a typhoon, fish seem to stay by the coral instead of going to open sea, as shown in Fig. 11b.

Although the scientific value of this finding has to be validated by marine biologists, it is easy to understand the asset that the proposed approach may provide to marine biology, especially in discovering and interpreting anomalous events.

As concerns the implementation of the algorithm, we have followed the workflow given in Algorithm 3.

Algorithm 3 Trajectory clustering algorithm

```
Get trajectories in standard water conditions and during typhoons
standardTrajs ← getStandardTrajectories()
typhoonTrajs ← getTyphoonTrajectories()
Apply k-means clustering
clusters ← kMeansClustering(merge(standardTrajs, typhoonTrajs))
Initialize counters to keep track of the number of trajectories associated to each cluster
clusterStandard[0, 1, ... clusters.size()-1] ← {0, 0, ...0}
clusterTyphoon[0, 1, ... clusters.size()-1] ← {0, 0, ...0}
Check which cluster each standard trajectory belongs to
for standardTraj in standardTrajs do
    clusterIndex ← getClosestCluster(standardTraj, clusters)
    clusterStandard[clusterIndex] ← clusterStandard[clusterIndex] + 1
end for
Check which cluster each typhoon trajectory belongs to
for typhoonTraj in typhoonTrajs do
    clusterIndex ← getClosestCluster(typhoonTraj, clusters)
    clusterTyphoon[clusterIndex] ← clusterTyphoon[clusterIndex] + 1
end for
```

5 Experimental results

The proposed system consists of different modules integrated together, therefore, the overall performance depends on the effectiveness of each of them.

5.1 Event detection

To evaluate the effectiveness of the typhoon detection module, we tested our method on 257 different videoclips taken from the *Fish4Knowledge* repository. These videoclips were sampled at 320×240 with a 24-bit color depth, at a frame rate of 6 *fps*. Each video was ten min long. Among all the 257 videoclips, 99 showed a typhoon event, and the remaining ones showed standard underwater conditions before and after the typhoon events. The videos of typhoon were selected among all the existing videos (recorded in the Taiwanese area during the last five years) by

Table 1 Obtained results in terms of CR and ER

K-Iteration	CR (%)	ER (%)
1	97.95	2.05
2	97.90	2.10
3	97.46	2.54
4	98.01	1.99
5	97.30	2.70

using the information (i.e. the dates) on the specific events gathered from DBpedia[2] and were then validated against two expert (over 20 years of experience) marine biologists.

For each video we processed 15 frames and the feature vector for each frame had dimensionality of [1×3910]. We selected only 15 frames per video since this was the best trade-off between computation costs and variability of the analysed scene: indeed by processing only 15 frames we were able to perform the method on-the-fly and, at the same, about one frame per minute was enough to grab the dynamics of a typhoon as the scenes change very gradually and slowly. This implies that the original dataset describing the features of all the 257 videos was a matrix of [3855×3910] values. After feature selection, carried out by means of PCA, the original dataset had dimensionality of [3855×18]. Then the reduced features vector was used to train and test the Support Vector Machines classifier. To calculate the expected risk since a large amount of data is considered, the K-fold cross-validation method was applied. By using $N = 3855$ labels for the data that represents the observations, we divide them into K subsets:

- K-1 subsets are used as training sets (learning);
- the remaining subset is used as the test set.

This operation is repeated leaving out each k subset, with $k = 1, 2, \cdots, K$ and the final risk is obtained from the combination of the k intermediate estimates. To evaluate the effectiveness of the classifier $K = 5$ was chosen, while Correct Rate (CR) and Error Rate (ER) were chosen to estimate the accuracy. Table 1 shows the obtained results in terms of CR and ER, while K varies from 1 to 5.

To test the capability of the proposed event detection approach in discriminating typhoon from other events, another evaluation was carried out by taking into account events "storm". In fact, the effects of a storm and of a typhoon on underwater scenes may appear similar at a visual inspection, as shown in Fig. 12.

In this case, we compared the "typhoon" videoclips (99) with 35 videoclips (extracted from the original dataset) depicting underwater scenes during storms. The K-fold (with $K = 5$) cross-validation method was also used. The achieved average correct rate was of 0.98, thus confirming the discrimination capability of the proposed method. The same procedure was applied to distinguish "storm" events from standard underwater conditions yielding an average correct rate (with $K = 5$) of about 0.91. This good performance indicates that a cascade of SVM classifiers is able to discriminate typhoon event, storm event from "not interesting" events (standard conditions).

[2] http://dbpedia.org/

(a) typhoon event (b) storm event

Fig. 12 Example of underwater scenes during **a** typhoon and **b** storm

The performance of the proposed system in detecting typhoon event was also estimated using the normalized detection cost (NDC) [26, 35], that represents the performance of detecting a specific event and it is defined as a weighted linear combination of missed detection (MD) and false alarm (FA) probabilities. The NDC for a specific event is given by:

$$NDC = C_{MD} \cdot P_{MD} \cdot P_T + C_{FA} \cdot P_{FA} \cdot (1 - P_T) \tag{6}$$

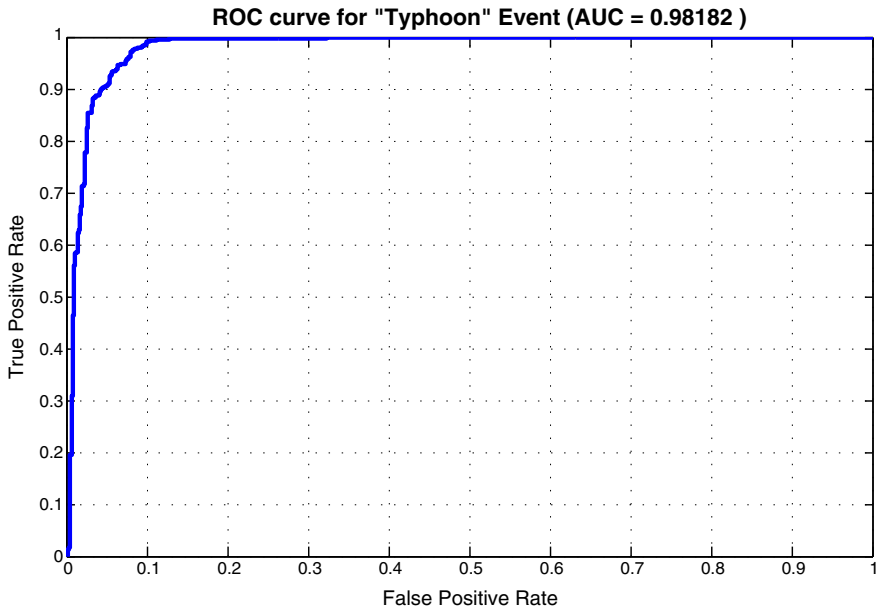
with $P_{MD} = \frac{N_{MD}}{N_T}$, $P_{FA} = \frac{N_{FA}}{N_T}$ that are, respectively, the missed detection and false alarm probabilities; where N_T , N_{MD}, N_{FA} are the numbers of videoclips, missed detections and false alarms regarding the specific event under investigation. P_T is the *a priori* rate of event instances E, whereas C_{MD} and C_{FA} are, respectively, the costs of MD and FA. The NDC was computed for both "typhoon" and "storm" events and we used 130 videoclips whose 52 with typhoons and 18 with storms as the training set. The evaluation set consisted of 127 videoclips, of which 47 with typhoons and 17 with storms. During our experiments, C_{MD} and C_{FA} were both set to 5. Table 2 reports the achieved values in terms of MD, FA and NDC for both "Typhoon" events and "Storm" events. Figure 13, instead, shows both the DET curve and the ROC curve of the SVM classifier when dealing with "Typhoon" events.

5.2 Fish trajectory extraction

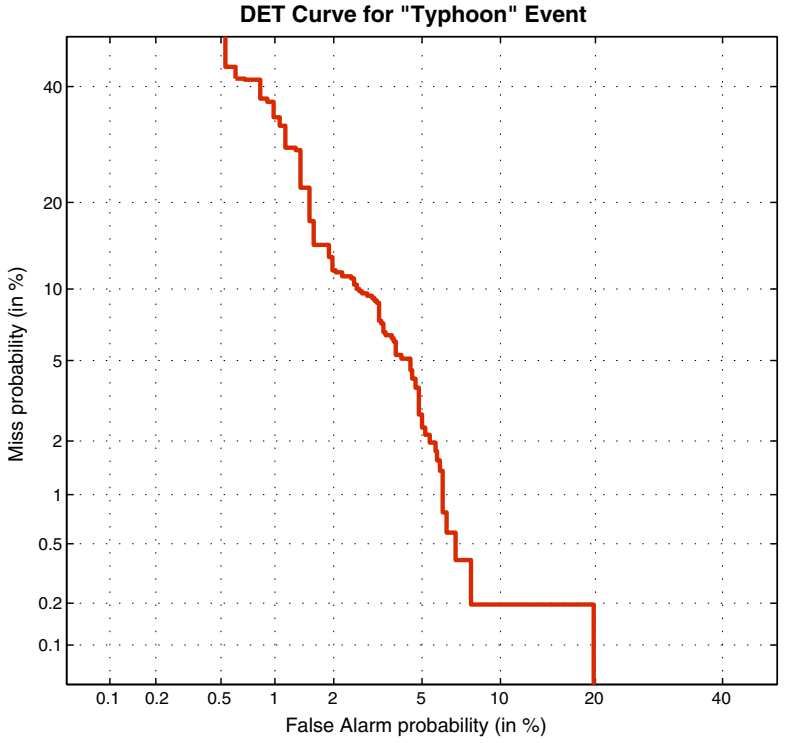
The fish trajectory extraction module was evaluated by assessing the performance of fish detection and tracking methods. As described in the "Fish Detection" section, we used four different algorithms for detecting fish, namely, Adaptive Gaussian Mixture Model ($AGMM$), Adaptive Poisson Mixture Model ($APMM$), Intrinsic Model (IM) and Wave-Back (WB), together with a post-processing module. To

Table 2 Evaluation results for events "Typhoon" and "Storm"

Events	E	N_T	MD	FA	P_{MD}	P_{FA}	P_T	NDC
Typhoon	47	127	2	7	0.02	0.05	0.37	0.19
Storm	17	127	3	3	0.02	0.02	0.13	0.10



(a) ROC Curve



(b) DET Curve

Fig. 13 Performance evaluation of the SVM Classifier for “Typhoon” events

evaluate the performance of the detection algorithms we adopted common metrics [35], i.e., detection rate (DR) and false alarm rate (FAR) which are defined as:

$$DR = \frac{N_{TP}}{N_{TP} + N_{FN}} \tag{7}$$

$$FAR = \frac{N_{FP}}{N_{TP} + N_{FP}} \tag{8}$$

where N_{TP}, N_{FP} and N_{FN} are, respectively, the number of true positives, false positives and false negatives. For the evaluation of the detection performance we used five videos (ten min long each) of the Fish4Knowledge repository, and two of them depicted typhoons. The videos had resolutions of 320×240 with a 24-bit color depth at a frame rate of 6 *fps*. The selection of these videos was based on specific features to test the effectiveness of every algorithm when non-standard conditions were encountered. In particular, the features taken into account were: dynamic backgrounds, illumination variations, high water turbidity, low contrast and camouflage phenomena. The ground truth on these videos was hand labeled by us using the Video Performance Evaluation Resource (VIPER) [15] and then validated by the same two marine biologists who labeled the ground-truth for typhoon event detection. The used videos are described in Table 3 together with the number of hand-labelled fish (N_F) in the ground truth data.

Moreover, in order to test the performance of the post-processing module, the results of the detection with and without the post-processing were estimated and are shown in Table 4. The results indicated that on average the best performance, in terms of fish detection in typhoon videos, was achieved by the Intrinsic Model combined with the proposed post-processing module and this was the method we used as basis for the subsequent fish tracking part.

For testing the tracking algorithm we used a different approach from ground-truth-based evaluation for two main reasons. First of all, the hand-labeling of ground truth for tracking is more tedious and more difficult than the detection one. The second reason is that it is not simple to define a quantitative performance metrics for an algorithm that may provide partially correct results, i.e. tracking algorithms provide multiple measures for the same object that represent the positions of all of its appearances and the algorithm may identify correctly only a subset of these positions.

Table 3 Description of the videos used as ground truth

Video	Description	N_F
1	Dynamic background Striped fish texture	156
2	Dynamic background Camouflage phenomena	1,373
3	Typhoon Frequent illumination variations Very low contrast	1,790
4	Typhoon Plants movements	34
5	High illumination Camouflage phenomena Striped fish texture	840

Table 4 Experimental results achieved when the detection algorithms run, respectively, without (NPP) and with (PP) the post-processing module

V.	Alg.	No post-processing					Post-processing				
		N_{TP}	N_{FP}	N_{FN}	DR	FAR	N_{TP}	N_{FP}	N_{FN}	DR	FAR
1	AGMM	102	38	54	0.65	0.27	129	11	27	0.83	0.08
	APMM	112	29	44	0.72	0.21	132	15	24	0.85	0.10
	IM	111	14	45	0.71	0.11	113	10	23	0.85	0.07
	WB	81	22	75	0.52	0.21	124	7	32	0.79	0.05
2	AGMM	957	89	416	0.70	0.09	1,126	87	247	0.82	0.07
	APMM	925	167	448	0.67	0.15	1,168	106	205	0.85	0.08
	IM	1,001	121	372	0.73	0.11	1,219	82	154	0.89	0.06
	WB	1,033	71	349	0.75	0.06	1,103	55	270	0.80	0.05
3	AGMM	1,253	228	537	0.70	0.15	1,458	159	332	0.81	0.10
	APMM	1,096	152	694	0.61	0.12	1,501	92	289	0.84	0.06
	IM	1,199	113	591	0.67	0.09	1,534	161	256	0.86	0.09
	WB	982	215	808	0.55	0.18	1,447	131	343	0.81	0.08
4	AGMM	26	10	8	0.76	0.28	30	6	4	0.88	0.17
	APMM	22	11	12	0.65	0.33	29	3	5	0.85	0.09
	IM	23	8	11	0.68	0.26	31	4	3	0.91	0.11
	WB	16	9	18	0.47	0.36	28	6	6	0.82	0.18
5	AGMM	563	60	277	0.67	0.10	686	68	154	0.82	0.09
	APMM	598	123	242	0.71	0.17	689	44	151	0.82	0.06
	IM	697	176	233	0.72	0.22	746	56	94	0.89	0.07
	WB	500	121	340	0.60	0.19	677	25	163	0.81	0.04

This is the case shown in Fig. 9, where we classified the two found trajectories (in four consecutive frames), respectively, as bad and good. Actually, the bad one indicates that the tracking algorithm followed the fish in three frames out of four, whereas the good one shows that the fish was correctly tracked in all its appearances. This consideration led us to use an online self-evaluation framework for the tracking algorithm based on the quality score q_S (described in Section 4.2.2) computed for assessing the quality of trajectories. Of course, such score is relative, since it is computed through software; nevertheless, the values computed by the self-evaluation algorithms [8, 20] have proved to provide reliable indication on the correctness of tracking. Let us recall that the quality score q_S lies in the range [0–1] and takes into account also the capacity of the algorithm to track fish when it leaves the scene and comes back in a reasonable interval of time (in the current version within six frames).

Therefore, the testing of the tracking algorithm involved the evaluation of the quality scores of the fish trajectories extracted from the same dataset (herein called D_1) used for event detection. However, for comparison purposes, the scores of the trajectories extracted from a wider set of 1,399 videos (D_2) were also computed. Table 5 shows the obtained results for both two datasets D_1 and D_2 in terms of minimum (m), maximum (M), average μ and standard deviation σ. The number of

Table 5 Obtained quality scores q_S for the datasets D_1 and D_2

Dataset	N_t	ρ_T	m	M	μ	σ
D_1	13,920	54.2	0.48	0.98	0.97	0.02
D_2	79,323	56.7	0.29	0.97	0.82	0.11

Table 6 Values of the parameters used for the normalization of the detection and tracking quality scores

Parameter	Value
α	75
β	$\frac{3}{500}$
γ	$\frac{5}{4}$
δ	1,600
ϵ	$\frac{3}{5}$
η	1.5
λ	100,00

extracted trajectories N_t and the average number ρ_T of trajectories per video are also shown for each dataset. It is possible to notice how the quality of the trajectories used for behavior understanding during typhoon (hence on dataset D_1) is even better than the one achieved on the larger dataset D_2.

The results confirm that both fish detection and fish tracking are not affected by the low quality of the typhoon videos, thus giving reliability to the considerations done in the previous section on fish behavior during typhoons.

As far as it concerns the detection post-processing module and the trajectory online evaluation method, in Sections 4.2.1 and 4.2.2 we described how quality scores are computed, without specifying, in the formulas, the exact values for the normalization parameters used to limit each value between 0 and 1. The choice of such parameters is particularly important, in the case of the detection quality scores, in order to make these values consistent with the threshold applied by the post-processing module, and has to deal with the issue of upperly unbounded quantities (e.g. Euclidean distances). The procedure we applied for setting such parameters for each feature follows:

- Initialize all parameters to a "neutral" value (i.e., 1 for multiplicative parameters and 0 for additive parameters).
- Process the videos in the test set in order to find the distribution of the quality values.
- Set the parameters so that, after the normalization, 95% of the values in the distribution fall into the [0, 1] range.

The values for the parameters which satisfy this property are shown in Table 6.

6 Concluding remarks

Keeping track of fish populations and analyzing their behavior in several different situations is a task which marine biologists struggle with, because of the difficulties in the collection of useful data and because of the typically used techniques that alter somehow the environment under observation. The Fish4Knowledge project aims at providing marine biologists with a tool for automatic fish detection, classification and behavior understanding, based on live video feeds from underwater cameras located in Taiwan's coral reefs. In this paper we introduced the architecture of one such system and described in detail the algorithms used for automatically detecting videos showing typhoons, and for fish detection and tracking, which are used as the basis for trajectory-based behavior understanding. Our approach for the detection of typhoon events is based on the extraction of multi-scale texture features from

the videos, which are then used to train a Support Vector Machine (SVM). We have shown how the resulting classifier is able to reach a very high accuracy (about 97%) in the discrimination between typhoon and non-typhoon videos. Based on these results, a trajectory clustering algorithm is used to study the differences in paths followed by the fish in standard underwater conditions and during typhoons. The comparison between the results obtained at those times gave indications on fish behavior, which of course have to be validated by marine biologists.

As future work, we are planning to extend the event detection algorithm to any event detection (as we simply did for "storm" events) problem which can be formulated in terms of texture description of videos. Indeed, the combination of texture features and classifier has been proved to be a reliable means. We are also aiming at improving both the detection and tracking algorithms (also adding some preprocessing steps [7]), which represent the foundation on which the entire system works, and the trajectories representation in order to better describe the 3D fish movements. Finally, in future all the achieved typhoon event data and the fish behavior events will be also mapped into RDF (Resource Description Framework) and published as Linked Open Data in order to provide sufficient detail to allow full replication of the experiments described, but also a sufficient level of abstraction to facilitate linking to existing event data.

In detail, we will allow third party applications to work directly on the raw data by providing a direct, lossless mapping from the relational tables to RDF [4]. As an example of a direct translation of a database row, one could directly address the fish detection table by dereferencing this URI: http://data.fish4knowledge.eu/fk4/direct/fish_detection/fish_id=256993, which would return in the following RDF (using Turtle notation):

```
@base <http://data.fish4knowledge.eu/fk4/direct>
@prefix xsd: <http://www.w3.org/2001/XMLSchema#> .
@prefix gml: <http://www.opengis.net/gml> .

# Example row of the fish_detection table:

<fish_detection/fish_id=260794>
  <fish_detection/fish_id> 260794 ;
  <fish_detection/video_id> <videos/video_id=3568> .
  <fish_detection/frame_id> 90 .
  <fish_detection/fish_id=260794>
  <fish_detection/detection_certainty> 0.77146 .
  <fish_detection/tracking_certainty> 1.0 .
  <fish_detection/timestamp> "2011-02-01"^^xsd:date .
  <fish_detection/bounding_box> "19,32 57,59"^^gml:posList .
  <fish_detection/countour_polygon> "55,34 56,34 61,34 ..."
    ^^gml:posList .
```

Second, to transform the event data to a higher abstraction we will map the typhoon event data and the fish behavior events to a number of common RDF event models such as F [50], SEM [67] and LODE [54]. These event-centric representations should also simplify aligning our data with other Linked Open Data, such as typhoon events that are part of DBPedia.

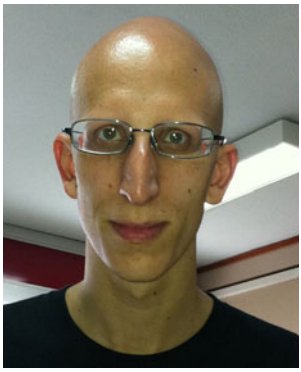
References

1. Albiol A, Silla J, Albiol A, Mossi J, Sanchis L (2009) Automatic video annotation and event detection for video surveillance. In: 3rd international conference on crime detection and prevention (ICDP 2009), pp 1–5
2. Ballan L, Bertini M, Bimbo AD, Seidenari L, Serra G (2011) Event detection and recognition for semantic annotation of video. Multimed Tools Appl 51:279–302
3. Benson B, Cho J, Goshorn D, Kastne R (2009) Field programmable gate array based fish detection using Haar classifiers. In: American academy of underwater science
4. Bertails A, Prud'hommeaux E (2011) Interpreting relational databases in the rdf domain. In: Musen MA, Corcho Ó (eds) K-CAP. ACM, pp 129–136
5. Bouaynaya N, Qu W, Schonfeld D (2005) An online motion-based particle filter for head tracking applications. In: Proc of the IEEE intl conf on acoustics, speech and signal processing
6. Brehmer P, Do Chi T, Mouillot D (2006) Amphidromous fish school migration revealed by combining fixed sonar monitoring (horizontal beaming) with fishing data. J Exp Mar Biol Ecol 334:139–150
7. Cannavo F, Nunnari G, Giordano D, Spampinato C (2006) Variational method for image denoising by distributed genetic algorithms on grid environment. In: Proceedings of the 15th IEEE international workshops on enabling technologies: infrastructure for collaborative enterprises. Washington, DC, USA, IEEE Computer Society, pp 227–232
8. Chau DP, Bremond F, Thonnat M (2009) Online evaluation of tracking algorithm performance. In: The 3rd international conference on imaging for crime detection and prevention
9. Cheung S-CS, Kamath C (2005) Robust background subtraction with foreground validation for urban traffic video. EURASIP J Appl Signal Process 2005(1):2330–2340
10. Chou H, Shiau Y, Lo S, Lin S, Lin F, Kuo C, Lai C (2009) A real-time ecological observation video streaming system based on grid architecture. In: HPC Asia 2009
11. Cline DE, Edgington DR, Mariette J (2008) An automated visual event detection system for cabled observatory video. In: VISAPP (1), pp 196–199
12. Comaniciu D, Meer P (2002) Mean shift: a robust approach toward feature space analysis. IEEE Trans Pattern Anal Mach Intell 24(5):603–619
13. Costa C, De Natale FGB, Granelli F (2004) Quality evaluation and nonuniform compression of geometrically distorted images using the quadtree distortion map. EURASIP J Appl Signal Process 2004:1899–1911
14. Dasiopoulou S, Mezaris V, Kompatsiaris I, Papastathis V-K, Strintzis M (2005) Knowledge-assisted semantic video object detection. IEEE Trans Circuits Syst Video Technol 15(10):1210–1224
15. Doermann D, Mihalcik D (2000) Tools and techniques for video performance evaluation. In: Proceedings 15th international conference on pattern recognition, 2000, vol 4, pp 167–170
16. Doucet A, De Freitas N, Gordon N (eds) (2001) Sequential Monte Carlo methods in practice. Springer Verlag
17. Edgington D, Salamy K, Risi M, Sherlock R, Walther D, Koch C (2003) Automated event detection in underwater video. In: OCEANS 2003. Proceedings, vol 5, pp 2749–2753
18. Elgammal A, Duraiswami R, Davis LS (2003) Efficient kernel density estimation using the fast gauss transform with applications to color modeling and tracking. IEEE Trans Pattern Anal Mach Intell 25:1499–1504
19. Elhabian S, El-Sayed K, Ahmed SH (2008) Moving object detection in spatial domain using background removal techniques—state-of-art. Recent Patents on Computer Science 1(1):32–54
20. Erdem C, Tekalp AM, Sankur B (2001) Metrics for performance evaluation of video object segmentation and tracking without ground truth. In: Proceedings of internation conference on image processing, vol 2, pp 69–72
21. Evans F (2003) Detecting fish in underwater video using the em algorithm. In: Proceedings of the 2003 international conference on image processing, ICIP 2003, vol 3, pp III – 1029–32, vol 2
22. Faro A, Giordano D, Spampinato C (2006) Soft-computing agents processing webcam images to optimize metropolitan traffic systems. In: Wojciechowski K, Smolka B, Palus H, Kozera R, Skarbek W, Noakes L (eds) Computer vision and graphics. Computational imaging and vision, vol 32. Springer Netherlands, pp 968–974. doi:10.1007/1-4020-4179-9-141
23. Faro A, Giordano D, Spampinato C (2011) Adaptive background modeling integrated with luminosity sensors and occlusion processing for reliable vehicle detection. IEEE Trans Intell Trans Syst 12(4):1398–1412

24. Faro A, Giordano D, Spampinato C (2011) Integrating location tracking, traffic monitoring and semantics in a layered its architecture. IET Intell Trans Syst 5(3):197–206
25. Forstner W, Moonen B (1999) A metric for covariance matrices. Tech rep, Dept of Geodesy and Geoinformatics, Stuttgart University
26. Gkalelis N, Mezaris V, Kompatsiaris I (2011) High-level event detection in video exploiting discriminant concepts. In: 9th international workshop on content-based multimedia indexing, Madrid, Spain (CBMI 2011)
27. Gordon N, Doucet A, Freitas N (1979) An algorithm for tracking multiple targets. IEEE Trans Autom Contr 24(6):843–854
28. Hariharakrishnan K, Schonfeld D (2005) Fast object tracking using adaptive block matching. IEEE Trans Multimed 7:853–859
29. Hearst M, Dumais S, Osman E, Platt J, Scholkopf B (1998) Support vector machines. IEEE Intell Syst Appl 13:18–28
30. Iqbal K, Abdul Salam R, Osman A, Zawawi Talib A (2002) Underwater image enhancement using an integrated colour model. AENG International Journal Of Computer Science 35(1): 31–41
31. Junejo IN, Foroosh H (2008) Euclidean path modeling for video surveillance. Image Vis Comput 26:512–528 (ACM ID: 1332292)
32. Khan Z, Gu I-H (2010) Joint feature correspondences and appearance similarity for robust visual object tracking. IEEE Transactions on Information Forensics and Security 5(3):591–606
33. Kuo C (2011) Damage to the reefs of Siangjiao Bay marine protected area of Kenting National Park, southern Taiwan during typhoon Morakot. Zoological Studies Environmental Biology of Fishes 50:457–462
34. Larsen R, Olafsdottir H, Ersbll B (2009) Shape and texture based classification of fish species. In: Image analysis. Lecture notes in computer science, vol 5575. Springer Berlin / Heidelberg, pp 745–749
35. Lazarevic-McManus N, Renno J, Jones GA (2006) Performance evaluation in visual surveillance using the f-measure. In: Proceedings of the 4th ACM international workshop on video surveillance and sensor networks, VSSN '06. New York, NY, USA, ACM, pp 45–52
36. Li W, Chen S, Wang H (2009) A rule-based sports video event detection method. In: International conference on computational intelligence and software engineering, 2009. CiSE, pp 1–4
37. Lowe D (2004) Distinctive image features from scale-invariant key-points. Int J Comput Vis 60:91–110
38. Morais EF, Campos MFM, Padua FLC, Carceroni RL (2005) Particle filter-based predictive tracking for robust fish counting. Brazilian Symposium on Computer Graphics and Image Processing 1:367–374
39. Nagashima Y, Ishimatsu T (1998) A morphological approach to fish discrimination. In: MVA98, pp xx–yy
40. Nanami A, Nishihira M (2002) The structures and dynamics of fish communities in an Okinawan coral reef: effects of coral-based habitat structures at sites with rocky and sandy sea bottoms. Environ Biol Fish 63:353–372. doi:10.1023/A:1014952932694
41. Nguyen H, Duhamel P, Brouet J, Rouffet D (2004) Robust vlc sequence decoding exploiting additional video stream properties with reduced complexity. In: IEEE international conference on multimedia and expo, 2004. ICME '04., vol 1, pp 375–378
42. Papadopoulos G, Mezaris V, Kompatsiaris I, Strintzis M (2008) Estimation and representation of accumulated motion characteristics for semantic event detection. In: 15th IEEE international conference on image processing, 2008. ICIP, pp 41–44
43. Porikli F (2005) Multiplicative background-foreground estimation under uncontrolled illumination using intrinsic images. In: Proc of IEEE motion multi-workshop
44. Porikli F (2006) Achieving real-time object detection and tracking under extreme conditions. J Real-Time Image Process 1(1):33–40
45. Porikli F, Wren C (2005) Change detection by frequency decomposition: wave-back. In: Proc of workshop on image analysis for multimedia interactive services
46. Porikli F, Tuzel O, Meer P (2005) Covariance tracking using model update based on lie algebra. In: Proc IEEE conf on computer vision and pattern recognition
47. Reid D (1979) An algorithm for tracking multiple targets. IEEE Trans Autom Control 24(6):843–854
48. Rouse W (2007) Population dynamics of barnacles in the intertidal zone. Marine Biology Research Experiment

49. Sankaranarayanan A, Veeraraghavan A, Chellappa R (2008) Object detection, tracking and recognition for multiple smart cameras. Proc IEEE 96(10):1606–1624
50. Scherp A, Franz T, Saathoff C, Staab S (2009) F–a model of events based on the foundational ontology DOLCE+DnS ultralight. In: Proceedings of the fifth international conference on knowledge capture KCAP 09. ACM, pp 137–144
51. Scherp A, Jain R, Kankanhalli M, Mezaris V (2010) Modeling, detecting, and processing events in multimedia. In: Proceedings of the international conference on Multimedia, MM '10. ACM, New York, NY, USA, pp 1739–1740
52. Schettini R, Corchs S (2010) Underwater image processing: state of the art of restoration and image enhancement methods. EURASIP J Adv Signal Process 2010:14:1–14:7
53. Shaish L, Levy G, Katzir G, Rinkevich B (2010) Coral reef restoration (Bolinao, Philippines) in the face of frequent natural catastrophes. Restor Ecol 18(3):285–299
54. Shaw, R, Troncy, R, and Hardman, L, (2009) LODE: linking open descriptions of events. In: Gómez-Pérez A, Yu Y, Ding Y (eds) ASWC, Lecture notes in computer science, vol 5926, Springer. pp 153–167
55. Sheng H, Li C, Wei Q, Xiong Z (2008) Real-time detection of abnormal vehicle events with multi-feature over highway surveillance video. In: 11th international IEEE conference on intelligent transportation systems, 2008. ITSC, pp 550–556
56. Shi J, Tomasi C (2008) Good features to track. In: Proc IEEE int conf comp vision and pattern recognition, pp 593–600
57. Sillito RR, Fisher RB (2009) Parametric trajectory representations for behaviour classification. In: BMVC
58. Siong Tew K, Han C-C, Chou W-R, Fang L-S (2002) Habitat and fish fauna structure in a subtropical mountain stream in Taiwan before and after a catastrophic typhoon. Environ Biol Fish 65:457–462. doi:10.1023/A:1021111800207
59. Soori U, Arshad M (2009) Underwater crowd flow detection using Lagrangian dynamics. Ind J Mar Sci 38:359–364
60. Spampinato C, Chen-Burger Y-H, Nadarajan G, Fisher RB (2008) Detecting, tracking and counting fish in low quality unconstrained underwater videos. In: VISAPP (2), pp 514–519
61. Spampinato C, Giordano D, Di Salvo R, Chen-Burger Y-HJ, Fisher RB, Nadarajan G (2010) Automatic fish classification for underwater species behavior understanding. In: Proceedings of the first ACM international workshop on analysis and retrieval of tracked events and motion in imagery streams, ARTEMIS '10. ACM, pp 45–50
62. Stauffer C, Grimson WEL (1999) Adaptive background mixture models for real-time tracking. In: Proceedings 1999 IEEE computer society conference on computer vision and pattern recognition Cat No PR00149, vol 2, no c, pp 246–252
63. Sugar CA, James GM (2003) Finding the number of clusters in a dataset. J Am Stat Assoc 98:750–763
64. Toyama K, Krumm J, Brumitt B, Meyers B (1999) Wallflower: principles and practice of background maintenance. In: The Proceedings of the seventh IEEE international conference on computer vision, 1999, vol 1, pp 255–261
65. Traiperm C, Kittitomkun S (2005) High-performance mpeg-4 multipoint conference unit. In: Proceedings of networks and communication system, pp 189–193
66. Tuzel O, Porikli F, Meer P (2006) Region covariance: a fast descriptor for detection and classification. In: Proc. 9th European conf on computer vision
67. Van Hage WR, Malais V, De Vries GKD, Schreiber G, Van Someren M (2012) Abstracting and reasoning over ship trajectories and Web data with the simple event model (SEM). Multimed Tools Appl 57(1):1–23
68. Varcheie P, Sills-Lavoie M, Bilodeau G-A (2010) A multiscale region-based motion detection and background subtraction algorithm. Sensors 10(2):1041–1061
69. Walther D, Edgington D, Koch C (2004) Automated video analysis for oceanographic research. In: Proc computer vision and pattern recognition, CVPR 2004, pp 544–549
70. Yilmaz A, Javed O, Shah M (2006) Object tracking: a survey. ACM Comput Surv 38(4): 1–45
71. Zhou S, Chellappa R, Moghaddam B (2003) Visual tracking and recognition using appearance-based modeling in particle filters. In: Proc intl conf on multimedia and expo
72. Zhou J, Clark C (2006) Autonomous fish tracking by rov using monocular camera. In: The 3rd Canadian conference on computer and robot vision, 2006. p 68

Concetto Spampinato received the Laurea in Computer Engineering in 2004, grade 110/110 cum laude, and the Ph.D. in 2008 from the University of Catania, where he is currently Research Assistant. His research interests include image and video analysis, content-based and semantic image and video retrieval, event detection in multimedia, medical image analysis, object detection and tracking in real-life environment. He has co-authored over 70 publications in international refereed journals and conference proceedings and he is member of IEEE and IAPR. He has also served in many international conference organization and technical program committee.



Simone Palazzo received the Laurea degree in Computer Engineering in 2010, grade 110/110 cum laude from the University of Catania, where he is currently doing his Ph.D. His interest activities include image and signal processing, image enhancement and reconstruction.

Bastian Boom received his bachelor engineer degree in Computer Science in 2002 from the Hogeschool van Amsterdam (www.hva.nl), The Netherlands. During this period he did two internships at fox-it. During the first internship he studied PGP (Pretty Good Privacy) and PKCS#11 (Smart card standard) and combined these two protocols in a C++ implementation (proof of concept) which was able to save the private key of PGP on a smartcard. During the second internship several strategies for Anomaly Detection on HTTP traffic were studied and he implemented a prove of concept of several methods for Anomaly Detection in Snort (Intrusion Detection Sofware). In 2005 he received the Master degree from the Vrije Universiteit (Free University) Amsterdam in Computer Science on a thesis entitled "Fast Object Detection". This thesis was the result of a succesfull internship at Prime Vision, where he did research at methods for fast detection (localisation) of license plates, faces and addresses in images. He has received this PhD at the University of Twente, in the fields of biometrics. His research topics are face detection, face registration and face recognition in video surveillance environments. He is at the moment working on the fish4knowledge project, at the University of Edinburgh. His research topic is fish recognition and clustering in large-scale video databases.

Jacco van Ossenbruggen is a senior researcher with the Interactive Information Access group at the Centrum voor Wiskunde en Informatica (CWI) in Amsterdam, and affiliated as an assistant professor with the Web & media research group at VU University in Amsterdam. His research interests include semantic Web interfaces for cultural heritage and other linked open data (working with Riste Gligorov, Alia Amin and Michiel Hildebrand on interfaces such as /facet) and Web-based data integration (working with Anna Tordai on alignment of SKOS vocabularies). Jacco is currently active in the Agora and PrestoPrime projects and the EuropeanaConnect Best Practice Network. Jacco obtained a PhD in computer science from VU University Amsterdam in 2001.

Isaak Kavasidis received the Laurea degree in Computer Engineering in 2009 from the University of Catania, where he is currently doing his Ph.D. His interest activities include semantic Web, RDF and semantic image processing.

Roberto Di Salvo received the Laurea degree in Computer Engineering in 2008, grade 110/110 cum laude from the University of Catania, where he is currently doing his Ph.D. His interest activities include mainly object detection and tracking in real-life scenarios.

Fang-Pang Lin is the Grid Application Division Manager in National Center for High Performance Computing (NCHC). He is one of key developers for developing a national cyber-infrastructure, namely Knowledge Innovation National Grid (KING). He initiated the Ecogrid project within PRAGMA as well as within KING to overarch international collaboration and extended development of KING for the year of 2003. His recent major efforts include Grid-based Lake Metabolism research and Telescience in PRAGMA, World Wide Metacomputing with HLRS, Germany and workflow model with AIAI, Edinburgh University, UK. He has been working on unstructured grid methods for Computational Fluid Dynamics (CFD) for more than a decade. His interests include multigrid methods, unstructured grid mesh generation, optimal design, multi-disciplinary numerical simulation and scientific visualization. His interest has been extended to advanced visualization, parallel and distributed computing and grid computing in recent years. Fang-Pang Lin obtained his PhD in University of Wales at Swansea, UK. He worked in Rolls-Royce University Computing Center in Oxford University as a research scientist after one-year postdoctoral research in Swansea. He joined NCHC in Oct., 1997 and has been working in numerical simulation and software engineering regarding application integration. He was the winner of 2006 Outstanding Achievement Award in Science and Technology, the Executive Yuan of Taiwan.

Daniela Giordano received the Laurea degree in Electronic Engineering, grade 110/110 cum laude, from the University of Catania, Italy, in 1990. She also holds the Ph.D. degree in Educational Technology from Concordia University, Montreal (1998). She is Associate Professor of Information Systems at the Engineering Faculty of the University of Catania since 2001. Daniela Giordano's research has been published in international refereed journals and conference proceedings and has dealt with topics such as: advanced learning technology, knowledge discovery, image processing, and information technology in medicine. Her current research interests include: cognitive systems, multimodal interaction and semantic technologies. She is a member of the IEEE and of the ACM.

Lynda Hardman is head of the Information Systems research cluster and a member of the Interactive Information Access research group at the Centrum Wiskunde & Informatica (CWI) in Amsterdam, The Netherlands. Her current research includes creating linked-data driven, user-centric applications for exploring media content and investigating user-centric interaction design in the context of developing technologies.

Robert B. Fisher has been an academic in the School of Informatics (originally in the former Department of Artificial Intelligence) at University of Edinburgh since 1984 and a full Professor since 2003. He received his PhD from University of Edinburgh (1987), investigating computer vision in the former Department of Artificial Intelligence. His previous degrees are a BS with honors (Mathematics) from California Institute of Technology (1974) and a MS (Computer Science) from Stanford University (1978). He worked as a software engineer for 5 years before returning to study for his PhD.